Goal of object-oriented programming

The goal/purpose of the *object-oriented design* is to decomposing an application into some number of modules. The most basic modules can individually complete some tasks, they are self-contained, means on them own, they can run without problem. Some more higher-level modules need to depend on the lower level of module to complete the task, so these modules are composed by other modules, they link some lower level modules to work together.

So OOP tries to break your large problem down into a number of smaller problems that are simpler and easier to handle and maintain. This approach is especially useful for enterprise level application. Java was designed from the ground up to be an object-oriented language, and all of the Java APIs and libraries are built around solid object-based design patterns.

Classes

So what's a class. We have already created a lot of classes in our previous class before. Every time we create a class with one file name same as the class name. this is how class loader knows what's the class and where to find the class. So far, we put only one single **main** method, which we mentioned is the entry point to run the class. Let's reiterate that **main** method is the only entry point that allow human to start the java program.
Classes are the building blocks of a Java application. A *class* can contain

- methods
- variables
- initialization code

Class is a blueprint for making class *instances*. Class is how you want to design something. Class instance, also called object, is created during program run (also called runtime). Instance/object is created based on the class.

e.g. you want to buy Nike shoes, hopefully everyone knows what's the shoe mold is, in the factory, in order to produce a large number of shoes, we have to have a shoe mold, so shoe with same style and size can be produced almost exactly same. So here shoe mold is the class, how we want the shoe shape like, then every produced shoe is an object or say instance. So class is how you want the object looks like, and object is the result from the class and created by JVM at runtime (runtime is when you code is running by the JVM, java virtual machine).

Shoe class example
https://github.com/JasonWang19/cacc_java/blob/master/week6/Shoe.java

1. create Shoe class
2. create the fields a shoe may have
3. create the methods a shoe may have

Fields are object properties, fields describe what kind of object it is going to be.
Methods are the functions/behaviors of an object. What an object can do.

Just like how you know a person, when the first time you meet someone, you will find out his/her properties like:

- gender
- height
- age
- name
- etc..

all of these are fields/properties of a person
then you can see what a person can do, e.g.

- play basketball
- play tennis
- play guitar
- programming java
- etc..

these are what a person can do/behavior

Object

There are multiple ways to create a new object from a class, but the simplest one is use **new.**

```
Shoe shoe = new Shoe();
```

So in this sample code, a shoe object is created/instantiated by using **new** key word. This means an object is created based on the design/blueprint of Shoe class.

Accessing Fields and Methods

Currently we may only need to remember them, and use default and public for now, and we will introduce detail when talk about package and inheritance later.

	Class	Package	Subclass (same package)	Subclass (diff package)	Outside Class
public	Yes	Yes	Yes	Yes	Yes
protected	Yes	Yes	Yes	Yes	No
default	Yes	Yes	Yes	No	No
private	Yes	No	No	No	No

Static Members

Static member vs instance member
Instance member only belong to certain created object/instance.
e.g. when we create a shoe, its size is 8, when we create another shoe, its size can be 9. They don't share the same size; the size only belongs to this particular shoe object.
Static member is shared by all of the objects from the same class.
e.g. ***TOTAL_USE_TIME***

is shared by all of the shoes instantiated from the Shoe class. Since simply put in our example, all the shoes have the same total life time, so static field is shared by all of the shoes.
Since **main** method is the entry point, so it has to be a static method.

Constants

final keyword is used to define a constant. Just as name indicated, value in constant cannot be changed.

Constructors

Constructors are a special method to create a new object. It is special in syntax, it doesn't have a return type, since the return type will always be the class of the object it is going to generate, so make sense, and the name of the constructor will also always same as the Class name. When not present, a default constructor will be created implicitly. But once constructor is explicitly defined, default constructor is no only available.
https://github.com/JasonWang19/cacc_java/blob/master/week6/Shoe2.java

Garbage Collection

Java uses a technique known as *garbage collection* to remove objects that are no longer needed. The garbage collector is Java's internal process, silently work in the background, stalking objects and awaiting their demise. It finds and watches them, periodically counting references to them to see when their time has come to destruct the object. When all references to an object are gone and it's no longer accessible, the garbage-collection mechanism declares the object *unreachable* and reclaims its space back to the available pool of resources. An unreachable object is one that can no longer be found through any combination of "live" references in the running application.

Homework

Create two classes called **Bag** and **Bird** by yourself, think about what fields and methods it may have.